XML (*Extensible Markup Language*) en quelques mots

Florence Clavaud

École nationale des chartes (Paris, France)

florence.clavaud@enc.sorbonne.fr

Cours d'introduction au stage de formation à TEI organisé à l'École des chartes, 9 mars 2011

De SGML à XML

- **XML** (*Extensible Markup Language*) est un héritier de **SGML** (*Standard Generalized Markup Language*), un **métalangage informatique pour le balisage du texte**, norme ISO (ISO 8879:1986) elle-même héritière d'un langage descriptif conçu en 1969 par des ingénieurs de la société IBM (Charles Goldfarb, Edward Mosher et Raymond Lorie)

- SGML était puissant et générique, mais aussi à la fois trop souple et complexe.
  Il a surtout été utilisé (et l'est parfois toujours) dans le cadre d'applications lourdes.
  Son utilisation la plus notable a été en fait... le langage **HTML** (HTML est né en 1989 ; dernière version : HTML 4.01, décembre 1999, également norme ISO/IEC 15445:2000) un langage pour la production d'hypertextes (pour le Web).

- Publication en **février 1998** par le W3C Consortium des spécifications du métalangage XML 1.0 (<http://www.w3.org/TR/1998/REC-xml-19980210>).
  Maîtres mots : **simplicité**, **clarté**, **universalité**.

- Pour comprendre l'histoire de la conception de XML, voir
  The Annotated XML Specification, de Tim Bray, l'un des pères de XML.

XML, un succès immédiat et important

- Les langages et outils permettant de produire, contrôler, échanger, transformer et exploiter des fichiers XML se sont multipliés, souvent associés à des normes du W3C : modèles de documents et de (méta)données, langages de modélisation, de présentation, de programmation, protocoles...

- Ce métalangage est donc aujourd'hui **utilisé partout en informatique**, qu'il s'agisse de structurer des informations dont la durée de vie est importante ou d'échanger des informations entre applications, qu'on ait à s'occuper essentiellement de données, essentiellement de documents, ou d'une combinaison des deux.

- Parmi les indices de popularité et de qualité de XML : HTML 4 a été reformulé conformément à XML, sous le nom XHTML 1.0, en janvier 2000 (révision en août 2002).

XML, depuis la version 1.0

- Depuis 1998, la norme XML a peu évolué.
  Dernière édition en date de la norme XML 1.0 : 5e édition
  (novembre 2008, cf. <http://www.w3.org/TR/2008/REC-xml-20081126/>).

- Une version 1.1 de la norme a été publiée en février 2004 (2e édition en août 2006, cf. <http://www.w3.org/TR/2006/REC-xml11-20060816/>), en particulier pour intégrer les évolutions d'Unicode ; mais XML 1.1 reste très peu utilisé.

Concepts : XML, pour baliser le texte

- **XML s'applique à tout type de texte.**
  Fondé sur le principe du **balisage** (qui consiste à isoler une portion de texte et à la marquer de manière explicite comme ayant un rôle ou étant de nature particulière), ce métalangage autorise l'***imbrication des balises*** et ***une granularité aussi fine que nécessaire***, dans la mesure où certaines règles syntaxiques simples sont respectées (on parle de ***document XML bien formé***).

- Un **modèle** (la liste des balises utilisables, leurs noms, les contraintes de leur emploi: position, cardinalité, type de contenu), ou grammaire, peut être défini, sous la forme d'une DTD ou d'un schéma, pour contrôler la qualité du document encodé (si un document XML est conforme à un modèle donné, on parle de ***document XML valide***).
  > Un bon modèle fournit de quoi baliser une catégorie de textes conformément à la nature intrinsèque de l'information, d'expliciter tout ce qui doit l'être, y compris la sémantique implicite.

Concepts : qu'est-ce qu'un texte ?

- Selon les concepteurs de la TEI en particulier, un texte n'est pas un document...
En quoi consiste l'essentiel d'un texte ?
    - en l'apparence des lettres et leur mise en page?
    - en la version originelle (prétendue) de cette copie?
    - en les interprétations/lectures apportées ou trouvées? en les intentions (supposées) de son auteur?

Un "texte" est quelque chose d'abstrait : la construction d'une communauté de lecteurs.
L'encodage explicite cette abstraction afin de la mieux gérer.

A lire sur cette question : la discussion qui a eu lieu entre novembre 1995 et juin 96 entre une trentaine de chercheurs sur le thème "Philosophy and Electronic Publishing: Theory and Metatheory in the Development of Text Encoding", ayant comme point de départ un texte de Allen Renear <http://philo.at/mii/mii/node7.html>

Concepts, suite

- XML permet d'identifier la structure d'un texte et d'informer sur la sémantique des informations qu'il contient, ou encore d'isoler des données.
  Rien n'empêche, bien sûr, d'utiliser XML pour spécifier la mise en forme d'un texte (cf. HTML). Mais par principe **la mise en forme (et le traitement) d'un document XML devraient relever de programmes externes** (parfois appelés feuilles de style).

- XML est conçu à l'ère de l'hypermédia : des liens peuvent être établis entre des sections du même document XML ou entre un document XML et d'autres ressources électroniques, qui peuvent ne pas être du texte.

Intérêt et enjeux : des informations structurées réutilisables

- Norme **simple**, sans ambiguité, permettant à un humain de lire l'information produite.

- Les fichiers XML sont des fichiers texte. XML supporte en outre un très grand nombre de systèmes d'écriture (il utilise le standard **Unicode** ; le codage par défaut de XML est UTF-8)

- Une norme universelle, concentrée sur le contenu informationnel, ne traitant pas de son utilisation, **indépendante des plates-formes informatiques**

- S'il y a un enjeu de pérennisation de l'information traitée, utiliser XML - et si possible une grammaire reconnue internationalement - pour structurer et **échanger** cette information est considéré comme une bonne pratique, et **les documents XML produits sont de bons candidats pour l'archivage pérenne** (cf. par ex. en France le Référentiel général d'interopérabilité, dont la version 1.0 a été publiée en mai 2009).

Intérêt et enjeux : des informations structurées réutilisables (2)

- **Accès, indexation et recherche de l'information améliorées** :
  - On peut avec certains outils informatiques indexer un (ou un ensemble de) document(s) XML en exploitant son (leur) marquage structurel, ce qui permet des recherches fines multi-critères et améliore considérablement la qualité des réponses par rapport à une recherche plein texte dans un document non structuré.
  - On peut aussi explorer un document XML en tant qu'arbre, en utilisant notamment le langage XPath, ou encore manipuler les noeuds de cet arbre à l'aide du langage DOM (*Document Object Model*).

- **XML, un format de stockage, pour autant de formats de diffusion qu'on le souhaite** :
  Après transformation par programme(s) en un ou plusieurs formats dédiés à un usage donné, le contenu structuré d'un document XML peut être consulté et diffusé sous forme imprimée (on peut transformer un document XML en fichier PDF par ex.), sous forme électronique (par ex. au format HTML...) ; ces **formats de sortie** peuvent être préparés à l'avance ou générés dynamiquement.
  Ceci fait de XML un **format pivot pour les nouvelles chaînes éditoriales.**

Intérêt et enjeux : des informations structurées réutilisables (3)

- **Même les éditeurs de logiciels de traitement de texte ont fait l'effort de définir des modèles XML** (Microsoft avec ISO/IEC 29500:2008 Office Open XML File Formats, et OpenOffice.org avec ISO/IEC 26300:2006 Open Document Format for Office Application). On peut parler, pour ces modèles, de langages de représentation.

- XML est aussi très souvent utilisé comme **format d'échange entre systèmes d'informations hétérogènes** (ex. : standard d'échange de données pour l'archivage, format utilisé pour les échanges entre machines au moyen du protocole OAI-PMH...).

Intérêt et enjeux : un investissement important, mais « rentable »

- « Passer à XML » a un coût...
  Cependant lorsqu'une communauté choisit et met en œuvre un ou plusieurs modèles XML pour ses besoins propres, elle peut :
  - constituer des collections homogènes de documents ;
  - répartir éventuellement la charge de leur production et de leur exploitation ;
  - gagner en autonomie par rapport aux logiciels informatiques ;
  - partager des outils et des savoir-faire ;
  - échanger ses fichiers

  bref gérer intelligemment son information documentaire

Le prologue d'un document XML

Deux composantes :

- la **déclaration XML** (optionnelle dans la version 1.0, d'usage recommandé)

Exemple :

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
```

Trois pseudo-attributs, indiquant la version de la norme XML, le système d'encodage utilisé pour le texte (par défaut UTF-8), l'existence d'une DTD exprimant certaines règles de contenu (`standalone` est optionnel ; valeur par défaut : `no`)

- la **déclaration de type de document** (optionnelle, utile et recommandée si le document est conforme à une DTD)

Exemple :

```
<!DOCTYPE ead SYSTEM "ead.dtd">
```

Un mot-clé obligatoire « DOCTYPE », suivi du nom de l'élément racine du document, suivi d'un mot-clé « SYSTEM », suivi entre guillemets de l'URL relative ou absolue du fichier de la DTD. Parfois, le mot-clé « PUBLIC » suivi d'un identifiant public, et, toujours, d'une URL. Parfois encore, entre crochets ([et])...., la DTD est en tout ou partie intégrée dans cette déclaration

Le corps du document XML : un arbre d'éléments

- Les **éléments** : **permettent de décomposer le texte en unités d'information**

  Exemple :

```
<titre>Le corps du document XML : un arbre
d'éléments</titre>
```

- La portion de texte ainsi isolée est marquée par une balise de début (**balise ouvrante** `<titre>`) et par une balise de fin (**balise fermante** `</titre>`)

- Tout élément a un **nom** (parfois appelé **identificateur générique**), ici c'est `titre`.

- Un élément peut contenir du texte et/ou d'autres éléments, ou être vide.

- Il existe **un et un seul élément englobant tous les autres, encodé en premier, juste après le prologue** : c'est l'**élément racine**.

Les attributs

Les attributs précisent la signification des éléments, leur ajoutent des caractéristiques

- Sont toujours saisis à l'intérieur de la balise ouvrante de l'élément
- Syntaxe : nomAttribut="valeur":

Exemple : `<date when="2011-03-09">9 mars 2011</date>`

- le nom de l'attribut obéit aux contraintes syntaxiques des noms XML
- la valeur est encadrée par des guillemets simples ou doubles.
- L'ordre des attributs n'est pas prescrit pour un élément donné
- Un attribut doit nécessairement avoir une valeur, même si cette valeur est nulle
- Pour un même élément, il ne peut y avoir deux attributs de même nom.

> Les attributs ne contiennent que du texte, peuvent donc servir pour stocker de l'information qui n'est pas elle-même structurée.

> En principe ne sont pas destinés à ajouter du contenu à l'élément, donc leur valeur n'est pas destinée à être affichée telle quelle dans un format de sortie. Mais ils pourront être exploités par des applications informatiques pour faire par exemple des index, ils peuvent servir à identifier les éléments (attributs de type ID), à établir des renvois (attributs de type IDREF).

Les noms des éléments (et des attributs)

- Règles pour l'écriture de ces noms (parfois appelés **identificateurs génériques**) :

  - la chaîne de caractères formant le nom peut contenir les caractères alphanumériques (lettres de a à z, et de A à Z, chiffres de 0 à 9, caractères non latins), le trait sous la ligne, le trait d'union et le point (à l'exclusion de tout autre signe de ponctuation ou blanc) ;

  - un nom ne peut pas commencer par un nombre, ni par un point, ni par un trait d'union ;

  - le nom ne peut pas commencer par "xml" (quelle que soit la casse utilisée).

- Attention, XML est sensible à la casse.

Autres composants : les entités

- **Les entités : des réserves de contenu, la plupart du temps à déclarer (dans la DTD) et utilisables n'importe où dans le document.**

  - Entités internes :

    - **entités caractères** prédéfinies ou non, pour saisir certains caractères que l'on doit obligatoirement coder (`<`, `>`, `&`, `&quot;`, `&apos;` : caractères codés au moyen des entités `lt`, `gt`, `amp`, `quot`, `apos`) ou difficiles à composer au clavier

    - **entités texte**, servant à donner un nom à des expressions ou phrases souvent répétées

  - Entités externes :

    - **texte** : fragments XML formant chacun un fichier

    - **non parsées** (fichiers non XML tels qu'images, enregistrements sonores…)

Un appel d'entité dans le corps du document XML se fait en utilisant l'esperluette (`&`), suivi du nom donné à l'entité, suivi du signe `;`

Autres composants

- **Les commentaires :**
  - Servent à... commenter, utiles pour documenter ce qu'on fait ou consigner des notes. Ne sont pas destinés aux programmes informatiques, qui les ignorent par défaut.
  - Commencent par `<!--` et se terminent par `-->`
  - Le double trait d'union (`--`) ne doit pas y apparaître.
- Les **instructions de traitement** :
  - Commencent par `<?` et se terminent par `?>`.
  - Définissent une cible (fournissent le nom de l'application à qui elles sont destinées, par ex.), et des arguments

  Ex. : `<?xml-stylesheet href="maCss.css" type="text/css" ?>`
  ou encore :
  `<?oxygen RNGSchema="schema/tei_all.rng" type="xml"?>`

  > Les 2 peuvent apparaître n'importe où, sauf dans une balise.

- **Sections CDATA** : sections de caractères non parsées. Commencent par `[!CDATA[` et se terminent par `]]`.

Espaces de noms

- de plus en plus souvent, afin de faciliter les échanges en évitant toute ambiguité, les noms des éléments et attributs sont rattachés à un **namespace** (**espace de noms**).
- Formellement un espace de noms est défini par son URI (**namespace-uri**).
- Si on utilise un ou plusieurs espaces de noms, l'espace de noms (ou les espaces de noms) utilisé(s) doivent être déclarés dans le fichier XML. On le fait le plus souvent en ajoutant un (ou plusieurs) attribut(s) xmlns à l'élément racine du fichier. Un attribut xmlns a pour valeur l'URI d'un espace de noms.
- L'espace de noms peut être déclaré comme l'espace de noms par défaut, dans ce cas l'URI n'est pas associée à un code donné et les noms des attributs et éléments ne sont pas préfixés.
- Un code peut en effet représenter l'espace de noms, il est alors utilisé à la place de l'URI pour préfixer attributs et éléments ; ce code est appelé préfixe (**namespace-prefix**).

Espaces de noms : exemples

```
- <TEI xmlns="http://www.tei-c.org/ns/1.0">
    <teiHeader><!-- autres éléments --></teiHeader>
    <text><!-- autres éléments --></text>
  </TEI>
```

Ici l'espace de noms par défaut est défini par l'URI http://www.tei-c.org/ns/1.0, c'est l'espace de noms TEI. Tous les éléments du fichier XML qui sont descendants de l'élément racine et qui ne sont pas préfixés sont dans cet espace de noms.

```
- <tei:TEI xmlns:rel="http://bibnum.bnf.fr/ns/reliure" xmlns:tei="http://www.tei-c.org/ns/1.0"
    xml:id="FR-751021006_RELIURE_NOT_00001">
<!-- le préfixe xml désigne l'espace de noms XML http://www.w3.org/XML/1998/namespace, qui n'a pas besoin d'être déclaré -->
    <tei:teiHeader><!-- autres éléments appartenant à l'espace de noms TEI ; leur nom est préfixé avec le préfixe tei --></tei:teiHeader>
    <tei:text>
      <tei:body>
        <tei:div tei:type="description">
          <rel:bookDescription rel:type="ms"><!-- bookDescription appartient, non pas à l'espace de noms TEI, mais à l'espace de noms http://bibnum.bnf.fr/ns/reliure -->
            <!-- autres éléments -->
          </rel:bookDescription>
        </tei:div>
      </tei:body>
    </tei:text>
</tei:TEI>
```

Règles à respecter pour avoir un document XML bien formé

Les principales règles sont les suivantes :

- concordance entre l'encodage du document et sa déclaration XML;
- existence des fichiers déclarés (déclaration de DTD, déclaration de fichiers entités externes) et concordance entre encodage des fichiers entités externes et leur déclaration XML;
- forme des appels d'entités ;
- présence de balises ouvrantes et fermantes appariées, imbrication des balises sans chevauchement ;
- respect des spécifications relatives aux noms XML (noms d'éléments et d'attributs) ;
- unicité des attributs dans un même élément, aucun attribut sans valeur
- forme des commentaires.

Le contrôle se fait par analyse syntaxique ou parsing (avec des outils appelés **parsers**).

Bien formé : oui ou non ?

- <seg>du texte</seg>
- <seg><foo>du</foo><bar>texte</bar></seg>
- <seg><foo>du <bar></foo> texte</bar></seg>
- <seg type="text">du texte</seg>
- <seg type='text'>du texte</seg>
- <seg type=text>du texte</seg>
- <seg type = "text">du texte</seg>
- <seg type="text">du texte<seg/>
- <seg type="text">du texte<gap/></seg>
- <seg type="text">du texte< /seg>
- <seg type="text">du texte</Seg>
- <seg type="text" type="toto">du texte</seg>

Modèles de documents

- Rappel : **définissent les contraintes que doit respecter une certaine classe de documents**
- Diverses syntaxes peuvent être utilisées pour écrire de tels modèles :
  - celle des **DTD** (partie intégrante de la norme XML 1.0), la plus utilisée encore aujourd'hui ;
  - celle, exprimée en XML, des **schémas XML** (norme du W3C publiée en mai 2001 ; 2e édition octobre 2004 ; voir <http://www.w3.org/XML/Schema>) ;
  - celle des **schémas RelaxNG** (norme ISO/IEC 19757-2 depuis 2003, amendée en 2008 ; voir <http://www.relaxng.org/>).

  Les schémas, qui peuvent s'écrire en XML quelle que soit la syntaxe choisie, permettent de contraindre plus fortement que les DTD le contenu des attributs et des éléments, et gèrent les espaces de noms.
- Un parseur sachant interpréter la syntaxe utilisée pourra vérifier qu'un document est conforme à un modèle donné (est valide).

  Exemples de parseurs : ceux intégrés aux éditeurs XML, aux navigateurs Web ; `xmllint` dans la librairie `libxml`, etc.

  Beaucoup d'applications XML sont validantes, ainsi un processeur XSLT n'opérera de transformation à partir d'un document XML associé à un modèle que si ce document XML est valide.

Quelques modèles XML publics

- Pour le **document structuré** :
  - un modèle générique couvrant les disciplines des SHS : TEI ;
  - pour la documentation technique : DocBook
- Des modèles pour représenter sous forme de texte :
  - des images vectorielles : SVG (Scalable Vector Graphics) ;
  - des formules mathématiques : MathML ;
  - des partitions musicales : MusicXML, plus récemment MEI (Music Encoding Initiative)
- Des **modèles de métadonnées** :
  - archivistiques : EAD (instruments de recherche archivistiques), EAC-CPF (notices d'autorité notamment pour les producteurs de fonds d'archives)
  - bibliothéconomiques : MarcXML, MODS (notices bibliographiques), MADS (notices d'autorité) ;
  - applicables à toute ressource : Dublin Core Metadata Element Set ;
  - spécifiques, dérivés du Dublin Core : TEF (pour les thèses : une norme AFNOR incluant un schéma XML W3C basé sur Dublin Core et sur METS) ; OLAC (un enrichissement du Dublin Core qualifié pour les ressouces linguistiques), ...
  - et encore : LOM et LOM-fr (pour les ressources pédagogiques) ; METS, un modèle de métadonnées descriptives, juridiques, administratives et structurelles pour des ensembles de fichiers électroniques, ...
- Modèles issus des travaux sur le **Web sémantique**: Resource Description Framework, d'où sont issus Dublin Core, SKOS pour les vocabulaires structurés, OWL pour les ontologies

Bibliographie très sélective

- Outre les ressources déjà mentionnées, parmi une très grande quantité de références, nous proposons :
  - L'article de Wikipédia en français sur XML, <http://fr.wikipedia.org/wiki/Extensible_Markup_Language>
  - *A gentle introduction to XML* : <http://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html>
  - *XML en concentré : manuel de référence* / Elliotte Rusty Harold & W. Scott Means; traduction de Philippe Ensarguet, Frédéric Laurent. - 3e éd. - Paris : O'Reilly, DL 2005. - 1 vol. (XX-760 p.) : couv. ill. en coul. ; 24 cm. - Index. - ISBN 2-84177-353-1 (br.) : 45 € - EAN 9782841773534 ;
  - L'espace XML francophone : actualités, discussions, articles et billets, sur le site Web <http://xmlfr.org/>